Н. А. Бурякова, А. В. Чернов (Ростов-на-Дону, РГСУ)**. Метод сокращенной верификации структур данных на основе компактного представления логических выражений диагностических операций.**

В данном докладе предлагается методика анализа таких структурных данных, как деревья, очереди, стеки, значительно ускоряющая их верификацию. Метод Model-checking проверяет на корректность такие операции для структурных типов данных, как вставка, удаление или поиск. Примерами подобных методик являются *Slam* [1], *Blast* [2], *Magic* [3]. Новая методика позволяет утверждать, что если операция O выполняется корректно на состоянии S, то гарантируется, что данная операция выполняется верно и на каждом из промежуточных состояний. Таким образом, однократная проверка корректности операции O гарантирует неповторяемость данного действия.

Верификацией устанавливается, что любая из операций вставки или удаления в структуре данных касается только одного пути в дереве от корня до листа. Тогда проверка осуществляется только для каждого из уникальных путей (и, возможно, ближайших соседей), а не для каждого уникального дерева. Число уникальных путей в каждом дереве является полиномом от n — числа полей в рассматриваемой структуре данных, что значительно ускоряет проверку в стом мысле, что все достижимые пути в дереве не обходятся

Алгоритмическое описание предлагаемого метода сокращенной верификации выглядит следующим образом: 1) выполняется проверка того, что операция либо завершится, либо выдаст сообщение об ошибке; 2) выполняется проверка корректности инварианта вычисления после выполнения императивной операции; 3) проверяется выполнение дополнительных условий, определенных пользователем системы (программистом либо оператором контроля системы).

Рассмотрим подробнее использование некоторых методов верификации в их программной реализации. Инвариантный класс структуры типа «очереди», описанный как *repOk*-метод в [4], добавляется в код программы в описании классов объектов. Он должен содержаться до или после каждого из *public*-методов в объектно-ориентированном описании системы. Метод *repOk* возвращает значение $true$, если объект является инвариантом рассматриваемого класса. Например, *repOk*-метод для структуры типа «стек» проверяет, что в связном списке нет циклов. Возможно использование локальных методов проверки $repOkLocal$. Локальный метод проверяет наличие частного экземпляра структурного класса данных и выдает значение $true$ на каждом узле при обходе дерева, построенного для целей верификации программной системы. Заметим, что любой глобальный инвариант может быть преобразован в локальный добавлением поля $blackHeight$ к каждому узлу.

В любой проверке, выполняемой по принципу Model-checking, которая проверяет структурные данные, должны определяться границы области поиска верифицируемых структур данных. В рассматриваемом нами методе в целом определяются следующие числовые характеристики верификации: максимальная рекурсивная вложенность путей обхода h дерева лексем верификации; максимальное число объектов каждого класса для всех объектов дерева.

Для компактного представления логических выражений и оптимизации пространства поиска используется аппарат BDD диаграмм (бинарные диаграммы принятия решений) [5], которые строятся следующим образом: каждый узел в BDD представляет один бит; сплошная линия от узла представляет бит, являющийся логической 1; пунктирная линия представляет собой отношения вида логический 0; для корня дерева логическая 1 представляет непустой указатель, 0 — пустой.

Незначащие элементы полей, соответствующие отсекаемым на дереве поиска ветвям, исключаются из рассмотрения. Поскольку изменять корневой узел в BDD нельзя, каждый раз при изменении BDD создается новая диаграмма BDD, копируя все узлы, стоящие выше места изменения, после чего продолжается поиск нового

места для отсечения ветвей на BDD. Сокращение происходит без явной проверки исключаемых состояний. В результате временные затраты на поиск компактного представления логического выражения для верификации являются линейными $O(n)$ по сравнению со стандартными методами верификации структур данных, имеющими вычислительную сложность $O(n^2)$.

СПИСОК ЛИТЕРАТУРЫ

1. *Ball T.*, *Majumdar R.*, *Millstein T.*, *Rajamani S.K.* Automatic predicate abstraction of C programs. — In: Programming Language Design and Implementation (PLDI), июнь 2001.
2. *Henzinger T. A.*, *Jhala R.*, *Majumdar R.* Lazy abstraction. — In: Principles of Programming Languages (POPL), январь 2002.
3. *Chaki S.*, *Clarke E.*, *Groce A.*, *Jha S.*, *Veith H.* Modular verification of software components in C. — In: International Conference on Software Engineering (ICSE), июнь 2003.
4. *Liskov B.*, *Guttag J.* Abstraction and Specification in Program Development. MIT Press, 1986.
5. *Bryant R. E.* Symbolic boolean manipulation with ordered binary decision diagrams. — ACM Computing Surveys, 1992, v. 24.